А. А. Варфоломеев

МЕТОД ХЕЛЛМАНА И ЕГО РАЗВИТИЕ ДЛЯ АНАЛИЗА НЕКОТОРЫХ КРИПТОСИСТЕМ В УСЛОВИЯХ МОДЕЛЕЙ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ[1]

Введение

Метод Хеллмана [1] является универсальным вероятностным методом нахождения ключей криптосистем и предполагает предварительный этап построения некоторой таблицы (таблиц) и оперативный этап нахождения ключа. Он ищет компромисс между временем работы и используемой памятью (time-memory trade-off, TMTO). Классический метод несколько раз был существенно модернизирован. История его развития имеет самостоятельный интерес.

Большое время предварительного этапа и объем требуемой памяти делали этот метод непрактичным, используемым в основном для теоретических оценок выбора параметров криптосистем. Ситуация меняется в условиях применения моделей распределенных вычислений, в том числе и с применением так называемых «облачных» вычислений и «облачным» хранением данных. Об актуальности этих методов кратко говорилось в работе [2]. Здесь мы рассмотрим эти вопросы более подробно.

Метод Хеллмана и его развитие

Напомним суть метода Хеллмана для произвольного блочного шифра. Преобразование открытого текста M_0 в шифртекст C_0 на ключе k_0 будем обозначать как $C_0 = E_{(k_0)}(M_0)$. Пусть далее все величины представляют собой двоичные векторы, и размеры M_0 и C_0 совпадают и равны L, а размер ключа равен n. Обозначим через $R = R_{(L,n)}$ отображение (редукции): $V_L \to V_n$. Определим функцию вида $f(k) = R(E_k(M_0))$.

Предварительный этап

Случайно выбираются m стартовых точек $SP_1, \ldots, SP_m$ из множества возможных ключей. Для выбранного значения t вычисляются значения матрицы размера (m Ч t) по правилу K(i, 0) = SPi, i = 1,…, m, K(i,j) = f(K(i, j – 1)), j = 1,…, t (называется матрицей Хеллмана). Конечные точки цепочек образуют последний столбец матрицы и имеют вид EPi = f (t) (SPi), i = 1,…, m. Только пары (SPi, EPi), i = 1,…, m, хранятся в памяти, отсортированные по EPi.

Оперативный этап

Применяя редукцию R к соотношению $C_0 = E_{k_0}(M_0)$, получим равенство: $R[C_0] = R[E_{k_0}(M_0)]$, правая часть которого равна $f(k_0)$ по определению функции $f(k)$. Обозначим левую часть через $Y_1 = R(C_0) = R(E_k(M_0)) = f(k_0)$. Значение Y_1 ищется среди значений конечных точек EP_i $(i = 1, \ldots, m)$. Если Y_1 равен какому-нибудь EP_i для какого-то значения i, то искомый ключ k_0 может быть равен $K_{(i,\,t-1)}$ или ключу, ему эквивалентному. Это следует из правила получения конечной точки $EP_i = f(K_{i,t-1})$, значение которой совпало с $Y_1 = f(k_0)$. Равенство $f(k_0) = f(K_{(i,t-1)})$ означает по определению функции равенство $R[E_{k_0}(M_0)] = R[E_{K_{i,t-1}}(M_0)]$. Если бы R было обратимым, то были бы получены бы равенства $E_{k_0}(M_0) = E_{K_{i,t-1}}(M_0)$ и $k_0 = K_{i,t-1}$. Если это не так, то надо проверить для $K_{(i,t-1)}$ выполнение равенства $E_{k_0}(M_0) = E_{K_{i,t-1}}(M_0)$. Для нахождения ключа $K_{(j,t-1)}$ используется стартовая точка SP_j. Если Y_1 не совпадает ни с одной из конечных точек и ключ не содержится среди элементов $(t-1)$-го столбца матрицы Хеллмана, то будем его искать среди ключей $(t-2)$-го столбца матрицы. Оставшиеся ключи в столбцах матрицы

[1] Работа выполнена в рамках мероприятия 1.2.1 Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 г. по направлению «Распределенные вычислительные системы».

Хеллмана проверяются аналогично путем вычисления $Y_s = f(Y_{s-1})$ для $s = 3, \ldots, t$. Ключ k_0 равен ключу $K_{(i,t-s)}$ для некоторого j тогда и только тогда, когда $Y_s = EP_i$ и $E_{K_{i,t-s}}(M_0) = C_0$.

Параметры метода

К параметрам относятся: N — размер пространства поиска; T_ρ — время предварительного этапа атаки; M — объем оперативной памяти; T — время оперативной работы атаки; D — количество данных, доступных атакующему на этапе оперативной работы. Вероятность успеха нахождения ключа можно сделать близкой к 1 с помощью генерации на предварительном этапе множества таблиц Хеллмана, выбирая для каждой уникальную функцию R. Число таких таблиц при описанных параметрах $m = t = N^{\frac{1}{3}}$ должно быть порядка $\left(N^{\frac{1}{3}}\right)$. При этом общее число операций оперативного этапа равно $T = \left(N^{\frac{1}{3}}\right) * \left(N^{\frac{1}{3}}\right) = \left(N^{\frac{2}{3}}\right)$. Потребуется память $M = \left(N^{\frac{2}{3}}\right)$. Соотношение для компромисса «время—память» имеет вид: $T(M^2) = N^2$ (кривая метода Хеллмана).

Новые предложения для построения таблиц

Во всех известных работах предлагается использовать различные функции вида $f_i = R_i(f(x))$ или $f_i = f(R_i(x))$, где R_i — функция модификации функции f. Если для f имеет место декомпозиция $f = f_p * f_{p-1} * \cdots * f_1$, где f_p зависит не от всех бит ключа, то в качестве R_i можно выбрать $f_\rho^{(-1)}$, что сократит время вычислений. Все $f_{(\rho-1)}, \ldots, f_1$ можно рассматривать как ограничение f на множестве ключей с фиксированными координатами, соответствующими выбору циклового ключа или функции R_i.

Метод характерных точек [3]

В этом методе цепочки ключей на предварительном этапе строятся не на фиксированную длину, а до тех пор, пока не будет найден ключ (точка) с характерной структурой. На оперативном этапе фактически не производится поиск по таблицам. Метод достаточно сложен для анализа [4].

Радужный метод

Метод [5] является развитием метода Хеллмана и позволяет теоретически сократить время оперативного этапа восстановления неизвестного параметра в 2 раза (на практике до 7 раз). Суть метода в использовании различных функций при построении цепочек ключей в матрице Хеллмана. Отсюда название: «радужный метод», «радужные цепочки», «радужные таблицы» («Rainbow method», etc.).

Чтобы получить и использовать функции f_i, надо хранить описание функций R_i, что требует памяти порядка t. Чтобы генерировать эти функции «на лету», предлагались разные способы. Например, в [6]: $f_i = f(x)$ XOR (X_i) для $i = 1, \ldots, t$, где вектора X_i являются состояниями линейного регистра сдвига максимального периода над GF(2).

Комбинация метода Хеллмана и радужного метода

В работе [7] предложена структура таблиц для предварительного этапа, которая является комбинацией структур таблиц Хеллмана и радужного метода. Кроме того, в ней метод реализован для D выходных значений однонаправленной функции. Предложенный метод дал новое соотношение между параметрами: $(T^3)(M^7)(D^8) = N^7$. Метод работает лучше других известных методов при $D > (N^{0,25})$. Таблица на предварительном этапе метода строится по следующему правилу: $K_{(i,0)}$, $i = 0, \ldots, m-1$; $K_{(i,j+1)} = f_j \pmod r K_{(i,j)}$, $j = 1, \ldots, rt$. Здесь $f_i = R_i * f$, где $f: V_n \to V_n$ является той однонаправленной функцией, для которой и надо найти прообраз. Элементы V_n называют точками. Пары начальных и конечных точек $(K_{(i,0)}, K_{(i,rt)})$, для $i = 0, \ldots, m-1$, хранятся в памяти отсортированными по второй компоненте. При $r = 1$ получается таблица Хеллмана, при $t = 1$ получается одна радужная таблица. В работе [7] также предлагается комбинация метода Хеллмана, радужного метода и метода характерных точек.

Применение метода Хеллмана для произвольной однонаправленной функции

Когда решается восстановление прообраза однонаправленной функции методом Хеллмана, то эта задача может ставиться для D известных значений однонаправленной функции. При этом будет требоваться нахождение любого одного прообраза. Это приводит к сокращению рассматриваемого множества прообразов и уменьшению трудоемкости предварительного этапа метода. При сокращении множества рассматриваемых прообразов предполагается, что при случайном равновероятном их выборе хотя бы один из искомых прообразов попадет в рассматриваемое множество.

Для каждой конкретной функции, возможно, существует более эффективный метод нахождения прообраза, например, при решении задачи дискретного логарифмирования.

Применение методов типа метода Хеллмана к поточным шифрам

В основном в литературе изучаются синхронные поточные шифры, в которых вырабатываемая последовательность обратимых отображений (гамма) не зависит от открытого и шифрованного текста. Рассматриваются следующие типы задач. Тип A: восстановить начальное состояние генератора выходной последовательности по отрезку выходной последовательности. Тип B: восстановить ключ по отрезку выходной последовательности генератора. При этом поточный шифр может использовать или не использовать так называемый открытый начальный вектор — НВ (начальный вектор — initial vector-IV).

Первыми работами, где применяется метод Хеллмана к поточным шифрам, являются работы [8, 9] для задач типа A. Пусть внутреннее состояние поточного шифра может принимать N возможных значений. Предполагается наличие у атакующего либо D различных выходных последовательностей (гамм) генератора ключевого потока длины log N, либо одной последовательности ключевого потока длины D + log N — 1. Задача — восстановить хотя бы одно внутреннее состояние генератора.

На стадии предварительной обработки выполняется следующее. Определим через f функцию, отображающую внутреннее состояние шифра размера в log N бит в выходной отрезок гаммы длины log N. Создается одна таблица, содержащая пары (s, f(s)), причем каждое значение s выбирается случайно. Таблица должна содержать M = N/D пар, отсортированных по второму значению. На стадии оперативной работы перебираются по очереди все D отрезков выходных гамм и ищется в таблице пара с совпадающей второй компонентой. Так как DM = N, то, по лемме о днях рождения, имеется хороший шанс найти одну такую пару в таблице. Соответствующая первая компонента пары и представляет искомое внутреннее состояние шифра. Из изложенного видно, что требуемая память M равна M = N/D. Временная сложность оперативного этапа T равна T = D. Кривая, связывающая параметры, для этого метода имеет вид TM = N при T_ρ = M (T_ρ — время предварительного этапа). Атака актуальна, если 1 <= T <= D. Например, можно выбрать T = M = D = $N^{1/2}$. Это лучше, чем соотношение в классическом методе Хеллмана $T(M^2) = (N^2)$, выполнимое при T = M = $N^{2/3}$.

Применение метода для поточных шифров более эффективно, чем применение его для произвольной однонаправленной функции. В работе [10] утверждается, что можно построить однонаправленные функции, противостоящие методу, для которого кривая баланса «время—память» имеет вид $T M^3 = N^3$ и T_P = N. Одним частным случаем, удовлетворяющим данной кривой, является выбор параметров вида T = M = $N^{3/4}$.

Для адаптации метода Хеллмана к поточным шифрам в работе [11] предлагается использовать тот же основной подход покрытия N точек (состояний) матрицами, определяемыми несколькими вариантами модификаций f_i функции f, которая задает отображение состояния в выходной отрезок гаммы. Функции f_i определяются соотношением $f_i(x) = h_i(f(x))$, где преобразование h_i заключается,

например, в перестановке бит значения f(x). Атака будет успешной, если любое из D значений выходного вектора будет найдено в любой из матриц. Поэтому можно уменьшить общее число точек, покрываемых матрицами, с N до N/D.

Это можно сделать двумя способами, либо уменьшая размер каждой матрицы, либо строя меньшее число матриц на предварительном этапе. Предлагается уменьшить число матриц с t до t/D. Это можно сделать, если t > D. Каждая матрица при этом потребует для хранения память размера m, но общая память уменьшится с M = mt до M = mt/D. Общее время предварительного этапа работы метода подобным образом сокращается с T_P = N до T_P = N/D. Для нахождения баланса «память—данные» используется то же соотношение $mt^2 = N$, для того чтобы оценить параметры m и t из различных соотношений. Время предварительного этапа равно T_P = N/D, в него эти параметры не входят. Время $T = t^2$, память M = mt/D и данные D удовлетворяют инвариантному соотношению $T M^2 D^2 = t^2 (m^2 t^2/D^2) D^2 = m^2 t^4 = N^2$. Это соотношение верно при t>=D, и поэтому $D^2 <= T <= N$. В частности, можно выбрать параметры $P = T = N^{2/3}$, $M = D = N^{1/3}$ или $T = M = N^{1/2}$ и $D = N^{1/4}$.

Другим методом, позволяющим распространить кривую Хеллмана вида $T M^2 D^2 = N^2$ на диапазон $T < D^2$, является метод из работы [12] (BSW-sampling). Этот метод может быть применен, если для некоторого легко достижимого подмножества выходных значений функции f входы, ведущие к этим выходам, могут быть эффективно перечислены. Предположим, например, что $N = 2^n$ и что множество A = {x: k младших бит значения f(X) равны нулю} может быть эффективно пронумеровано. В этом случае атакующий рассматривает на предварительном этапе только значения x из множества A.

С каждым таким значением x он связывает два (n-k)-битовых вектора: короткое имя, которое используется в процедуре перенумерации для определения x, и выходной вектор, который соответствует (n-k) наиболее значимых бит значения f(x). Затем рассматривается функция g: $\{0, 1\}^{n-k} \to \{0, 1\}^{n-k}$, определяемая как g: (сокращенное наименование) = имя соответствующего выхода. Для этой функции создаются таблицы Хеллмана. На оперативном этапе атаки рассматриваются только точки вида y из f (A) и используется обращение g, чтобы найти прообраз одной из них.

В результате данные, доступные атакующему, снижаются с D до $D/(2^k)$, в то время как инвертируемая функция, отображающая n-битный вход в n-битный выход, изменяется на функцию, отображающую (n-k)-битный вход в (n-k)-битный выход. Как показано в [12], кривая Хеллмана для описанного метода имеет вид $N^2 = T(M^2)(D^2)$, область применимости кривой увеличивается до $T \geq (2^{-k}D)^2$.

В работе [13] рассматривается задача типа B. Атакующий не знает заранее HB и не может использовать его на предварительной стадии метода. Возможный подход в данном случае заключается в том, чтобы рассматривать HB как часть секретного ключа. Но тогда существенно увеличивается размер ключа. Предлагается выбрать заранее некоторое множество HB и попытаться решить задачу для выбранных HB. Этот метод похож на метод из работы [12] для функции f: (Ключ, HB) → (выходной вектор), на множестве из 2^{k+v} значений. Так как HB считается известным, то легко перечислить множество выходных векторов, соответствующих входам вида.

В работе [14] представлено описание атаки типа «угадать-и-определить» на поточные шифры, которая основывается на угадывании части внутреннего состояния и нахождении оставшейся неизвестной части состояния шифра по известным битам выходной гаммы. Показано, что основная атака всегда может быть распространена до атаки на основе балансировки «времени — памяти — данных» атаки. Это позволяет улучшить оперативный этап атаки по памяти, времени предварительной обработки и данным.

Модели распределенных вычислений

Проще всего рассматривать стационарные модели, для которых значения параметров, описывающих вычислительную систему, не меняются в ходе реализации алгоритма. По способу организации вычислений различают модели с координатором и без него. Координатор раздает задания отдельным узлам сети и объединяет результаты вычислений. Обобщенное описание параметров моделей распределенных вычислений следующее: $N(t)$ — число активно работающих процессоров (вычислителей) в момент времени $t = 1,2,\ldots$; ϖ_i — производительность процессора i-го вычислителя (эл.оп./сек), $i = 1,\ldots,N(t)$; μ_i — объем памяти вычислителя i-го участника (яч); v_i — производительность записи/считывания единицы информации из внешней памяти i-го вычислителя (яч/сек); χ_i — пропускная способность канала передачи данных i-го вычислителя (яч/сек). Как следует из приведенного описания методов типа метода Хеллмана, для нахождения ключей не требуется интенсивного обмена информацией между узлами.

Описание реализации метода Хеллмана в условиях модели распределенных вычислений

Помимо приведенных выше параметров модели вычислений необходимо учитывать параметры, связанные с особенностями самой решаемой задачи, например такие параметры: cj — вычислительная сложность обработки j-го ключа, j = 1,…,K; ρj — вероятность использования j-го ключа, j = 1,…,K. Наиболее распространенным является предположение о равновероятности и независимости каждого ключа, а также об одинаковой сложности обработки каждого ключа, т. е. предположение, что ρj = 1/K, cj = c = const. Если это не имеет места, то алгоритм решения может быть изменен.

Как следует из приведенных выше описаний методов, на предварительном и оперативном этапах работы можно использовать распределенные вычисления. Основным способом распараллеливания вычислений является выбор начальных точек цепочек ключей для построения таблиц.

При неограниченной памяти у координатора он хранит результаты предварительного этапа и рассылает каждому участнику данные о цепочках для обработки на оперативном этапе. При этом предварительный этап получения начальных и конечных точек цепочек может выполняться участниками как независимо (что увеличивает трудоемкость примерно в $log(K)$ раз), так и при задании начальных точек координатором. Общая производительность сети при стационарной модели вычислений равна $\pi_\Sigma = \sum_{i=1}^{N} \pi_i$. Именно эта производительность и определяет выигрыш во времени решения задачи нахождения ключа по сравнению со временем одного процессора.

В условиях ограниченности памяти у координатора часть участников вычислений будет задействована в получении пар точек для цепочек предварительного этапа и отправке их координатору для рассылки участникам оперативного этапа.

В задаче определения прообраза для однонаправленной функции, помимо приведенных выше общих параметров модели вычислений, особо важную роль могут играть следующие параметры: c_j — вычислительная сложность обработки j-го прообраза, $j = 1,\ldots,K$; ρ_j — вероятность использования j-го прообраза, $j = 1,\ldots,K$. Здесь K — число возможных прообразов (или их рассматриваемая часть). В связи с неравновероятностью использования прообразов надо на предварительном этапе каждой цепочке поставить в соответствие суммарную вероятность прообразов, ее составляющих. Точно так же каждой цепочке ставится в соответствие и суммарная трудоемкость обработки составляющих ее прообразов. При этом координатор определяет баланс вероятности цепочки и трудоемкости ее обработки на оперативном этапе.

Как и в других задачах, на выигрыш от распределенных вычислений влияет общая производительность сети. Принцип распараллеливания, как следует из приведенного описания методов, остается тем же — по начальным точкам получаемых на предварительном этапе цепочек. Например, при N одинаковых машин, используемых в распределенных вычислениях, выигрыш может достичь N раз.

Вывод

Все методы типа метода Хеллмана для балансировки времени оперативной работы нахождения неизвестных параметров криптосистем, используемой памяти и количества данных могут быть эффективно реализованы с помощью распределенных вычислений. Использование распределенных вычислений позволяет достигать указанными методами для различных криптографических примитивов не только теоретических, но и практических результатов.

СПИСОК ЛИТЕРАТУРЫ:

1. *Hellman M.* A Cryptanalytic Time – Memory Trade-Off. IEEE Trans. on Information Theory. Vol. 26. P. 401–406.
2. *Варфоломеев А. А., Когос К. Г., Коренева А. М., Фомичев В. М.* О сложности реализации некоторых алгоритмических методов криптоанализа с помощью распределенных вычислений // Безопасность информационных технологий. 2011. № 4. С. 37–43.
3. *Hong J., Jeong K. C., Kwon E. Y.* Variants of the Distinguished Point Method for Cryptanalytic Time Memory Trade-Offs. IACR Eprint Report 2005/090, 2005. http://eprint.iacr.org/2008/054 .
4. *Hong J., Moon S.* A Comparison of Cryptanalytic Tradeoff Algorithms. IACR Eprint Report 2005/090, 2005. http://eprint.iacr.org/2010/176 .
5. *Oechslin P.* Making a Faster Cryptanalytic Time-Memory Trade-Off. CRYPTO 2003, LNCS, vol. 2729, (Springer, 2003). P. 617–630.
6. *Mukhopadhyay S., Sarkar P.* Application of LFSRs in TMTO Cryptanalysis. WISA 2005, LNCS 3786. Springer-Verlag, 2006. P. 25–37.
7. *Mukhopadhyay S., Sarkar P.* On the Effect of TMTO & Exhaustive Search. IWSEC 2006, LNCS 4266. Springer-Verlag, 2006. P. 337–352.
8. *Golic J. Dj.* Cryptanalysis of alleged A5 stream cipher. Eurocrypt 1997. LNCS. Springer-Verlag, Heidleberg (1997). Vol. 1233. P. 239–255.
9. *Babbage S.* Improved exhaustive search attacks on stream ciphers. European Convention on Security and Detection, IEE conference publication. 1995. № 408. P. 161–166.
10. *Fiat A., Naor M.* Rigorous Time/Spase Tradeoffs for Inverting Functions. SIAM J. Comput. 29(3), 790–803 (1999).
11. *Biryukov A., Shamir A.* Cryptanalytic TMDTO for Stream Ciphers. LNCS 1976, ASIACRYPT 2000. P. 1–13. Springer-Verlag.
12. *Biryukov A., Shamir A., Wagner D.* Real Time Cryptanalysis of A5/1 on a PC. FSE 2000, LNCS 1978, (Springer, 2001). P. 1–18.
13. *Hong J., Sarkar P.* Rediscovery of Time Memory Tradeoffs. IACR Eprint Report 2005/090, 2005. http://eprint.iacr.org/2005/090.
14. *Chew G., Khoo K.* A general framework for guess-and-determine and TMD. 2007. http://www.techrepublic.com/whitepapers/a-general-framework-for-guess-and-determine-and-time-memory-data-trade-off-attacks-on-stream-ciphers/4410741